UNCLASSIFIED

Australian Government
Department of Defence
Defence Science and
Technology Organisation

The Need for Software Architecture Evaluation in the Acquisition of Software-Intensive Systems

Liming Zhu[a], Mark Staples[a] and Thong Nguyen[b]

[a]National Information and Communication Technology Australia, (NICTA)

[b]Aerospace Division, Defence Science and Technology Organisation

DSTO-TR-2936

ABSTRACT

The software architecture for a software-intensive system defines the main elements of the system, their relationships, and the rationale for them in the system. Software architecture is fundamental to whether a system can achieve its quality objectives. Architecture evaluation is an approach for assessing whether a software architecture can support the system needs, especially its non-functional requirements (also known as quality requirements). Architecture evaluation can be used at different stages of a project, and is an effective way of ensuring design quality early in the lifecycle to reduce overall project cost and to manage risks.

This report describes software architecture and architecture evaluation, and summarises some of the key benefits for software architecture evaluation that have been observed both in industry and in international Defence contexts. We make some general recommendations about architecture evaluation in the context of Australian defence acquisition.

RELEASE LIMITATION

Approved for public release

UNCLASSIFIED

Published by

Aerospace Division
DSTO Defence Science and Technology Organisation
506 Lorimer St
Fishermans Bend, Victoria 3207 Australia

Telephone: 1300 333 362
Fax: (03) 9626 7999

AR 015-849
January 2014

The Need for Software Architecture Evaluation in the Acquisition of Software-Intensive Systems

Executive Summary

Australian Defence and the local defence industry are facing new challenges from ever-larger, more complex and interconnected software-intensive systems. Quality problems arising in the development of such systems are often only recognised late during development, leading to significant rework, and cost and schedule overruns. The risks for such problems can form at the start of an acquisition process, and at the earliest stages of high-level system design.

The *software architecture* for a software-intensive system defines the main elements of the system, their relationships, and the rationale for them in the system. Software architecture is fundamental to whether a system can achieve its quality objectives. *Architecture evaluation* is an approach for assessing whether a software architecture can support the system needs, especially its *non-functional* requirements (also known as quality requirements). Architecture evaluation can be used at different stages of a project, and is an effective way of ensuring design quality early in the lifecycle to reduce overall project cost and to manage risks.

This report describes software architecture and architecture evaluation, and summarises some of the key benefits for software architecture evaluation that have been observed both in industry and in international Defence contexts. We make some general recommendations about architecture evaluation in the context of Australian defence acquisition.

Recommendation 1: Incorporate architectural requirements into the Request for Tender package.

Recommendation 2: Integrate architecture evaluation activities across the acquisition process.

Recommendation 3: Build architecture capabilities in Australian defence industry.

Recommendation 4: Provide appropriate contractor incentives for architecture definition, analysis, and evaluation throughout the project lifecycle.

Acknowledgements

The authors would like to express our appreciation to the Defence Materiel Organisation for providing funding for this research and to Mr Adrian Pittman for supporting this work.

Authors

Dr Liming Zhu
NICTA

Dr. Liming Zhu is a research laader in the Software Systems Research Group at NICTA. NICTA is Australia's Information and Communications Technology Research Centre of Excellence. Dr Zhu formerly worked in several technology lead positions in the software industry before obtaining a PhD in software architecture at the University of New South Wales (UNSW). He is Australia's expert delegate to the ISO/SC7 Working Group 42 on architecture related ISO standards. His research interests include software architecture, service engineering, model driven development and business process modelling.

Dr Mark Staples
NICTA

Dr. Mark Staples is a principal researcher in the Software Systems Research Group at NICTA. NICTA is Australia's Information and Communications Technology Research Centre of Excellence. Prior to NICTA, he worked in industry in verification and process management roles for the development of safety-critical transportation systems and financial transactions infrastructure systems. Dr Staples has a PhD in computer science from the University of Cambridge, and completed undergraduate degrees in computer science and cognitive science at the University of Queensland. His research in software engineering has included topics in software architecture, software process, software configuration management, software product line development, and formal methods.

Dr Thong Nguyen
Aerospace Division

Dr Thong Nguyen is the project science and technology advisor to project AIR5440 and was the research manager in Mission Systems Integration Branch, Aerospace Division. Dr Nguyen has been conducting research in different fields including insect vision based motion detection, Data Fusion, Situation and Threat Assessment, Automation, Software Architecture, and Systems Integration Risk Assessment, and Systems Engineering. He has published over 50 research papers including book chapter, journal and conference papers. He has provided technical advice to various acquisition projects. He also provided advice to DMO on how to introduce Software Architecture evaluation into earlier stages of the acquisition process in order to identify and mitigate architecture-related technical risks as early in the acquisition process as possible. He was posted to the Wedgetail IPT in Seattle for two years as a technical advisor on Situation and Threat assessment and Sensor Management. He has 17 years of experience in using science and multidisciplinary research to enhance Defence Capability and holds a BE (Honour I) in Computer Systems Engineering, a BSc in Applied Mathematics, and a PhD in Electronics Engineering, all from Adelaide University; and a MBA in Technology Management from Deakin University.

Contents

This page is intentionally blank

Glossary

ADF	Australian Defence Force
ATAM	Architecture Trade-off Analysis Method
COTS	Commercial-Off-The-Shelf
DMO	Defence Materiel Organisation
DSTO	Defence Science and Technology Organisation
FPS	Function and Performance Specification
GIG	Global Information Grid
ISO	International Standard Organisation
MDA	Model Driven Architecture
MDD	Model Driven Development
MOTS	Military-Off-The-Shelf
NCW	Network Centric Warfare
NICTA	National ICT Australia
OCD	Operational Concept Document
RFP	Request for Proposals
RFT	Request for Tenders
SEI	Software Engineering Institute

This page is intentionally blank

1 Introduction

Large and complex software-intensive projects often exhibit quality problems in late stages, may not be able to deliver on promised functionality on time, and thus require significant rework late in their schedule (Boehm, Valerdi et al. 2008). These issues affect not only the developers of software-intensive systems but also acquirers. The risks for such problems can form at the start of an acquisition process, and at the earliest stages of high-level system design. The *software architecture* for a software-intensive system defines the main elements of the system, their relationships, and the rationale for them in the system. Software architecture is fundamental to whether a software-intensive system can achieve its quality objectives, and is also important in structuring the system decomposition and associated work breakdown structure.

Architecture evaluation is an approach for assessing whether a software architecture will be complete and consistent in terms of the system needs, especially the *non-functional* requirements (also known as quality requirements). Architecture evaluation can be used at different stages of a project, and is an effective way of ensuring design quality early in the lifecycle to reduce overall project cost and to manage risks. For example, AT&T, Avaya, Lucent and Millenium Services have reported around 700 architecture evaluations between 1998 and 2005. They estimate that projects of 100,000 non-comment source lines of code have saved an average of US$1 million each by using software architecture evaluations for identifying and resolving problems early (Maranzano, Rozsypal et al. 2005). Another recent survey of the state of practice in architecture evaluation among 235 industry participants identified a range of benefits – the top five being: architecture evaluation's ability to identify project risks (88% responses), assess quality (77% responses), identify reuse opportunities (72% responses), promote good practices within an organisation (64% responses) and reduce project cost by reducing design defects (63% responses) (Babar and Gorton 2009). Architecture evaluations can also help to identify best practices in projects and socialize such practices across an organization, thereby improving the organization's quality and operations (Maranzano, Rozsypal et al. 2005).

The Australian Defence Force (ADF) sponsors a large number of software-intensive system acquisition projects. Like many other systems, modern weapon systems are becoming more software-intensive, larger, more complex, and with more networked capabilities (DoD 2009). The focus in the ADF is also shifting from technology readiness (i.e. maturity and feasibility) of individual technologies to the technical risk associated with systems and their *integration* with other systems (Smith, Egglestone et al. 2004) rather than new development. Industry adoption of architecture evaluation has emerged out of the need to deal with large-scale integration projects.

The global defence industry has also undergone significant consolidation over the last several decades, resulting in an industry dominated by a few very large defence companies, mostly based in Europe and North America. Many acquisitions projects are encouraged to use Off-The-Shelf (OTS) systems including Military-Off-the-Shelf (MOTS), Government-Off-The-Shelf (GOTS) or Commercial-Off-the-Shelf (COTS) systems with limited customisation. However, the use of such OTS systems poses significant risks to quality, schedule and sustainment due

to reduced control and exposure to long supply chains and dispersed global manufacturing, especially when local industry capability is lacking (DoD 2010). Architecture evaluation techniques can accommodate analysis of black-box OTS software.

As outlined in the Defence white paper (DoD 2011), the specialisation entailed in complex projects has created new opportunities and dependencies in the global defence industry. The complexity of systems development and integration involved in modern defence projects has led to new partnerships and networks between prime suppliers and their suppliers and contractors. This is an opportunity for Australian defence companies to become part of the global supply chain of the primes. Explicit and well-managed architecture in major systems will assist Australian SMEs to identify opportunities for developing components that integrate into the global supply chains of international primes.

Similar issues have also been identified in the US Department of Defense (DoD). It was clearly observed that past acquisition evaluation methods (DoD 1985; DoD 1994) were inadequate (Hantos 2004) due to their lack of focus on quality attributes and integration challenges. The response to this was to introduce a focus on architecture-driven system acquisition. Architecture-related activities have been tightly integrated into the RFP language (Bergey and Fisher 2001; Bergey, Fisher et al. 2002) and has produced positive results (Bergey and Morrow 2005).

In this paper, we introduce the key concepts in software architecture and architecture evaluation. We survey the academic and industry state-of-the-art and state-of-practice for architecture evaluation. We review the architecture evaluation practices in US DoD system acquisition. Finally, we point to some of the possibilities to integrate architecture evaluation in Australian defence acquisition.

2 The Role of Software in Modern Complex Systems

Software is rapidly becoming the centrepiece of complex system design. A large portion of complex system functionalities are now achieved through software. For example, in a period of forty years, the percentage of functionality provided by software to pilots of military aircraft has risen from 8% in 1960 (F-4 Phantom) to 80% in 2000 (F-22 Raptor) (Ferguson 2001). As a consequence of software's growing complexity and its critical role in modern systems, software architecture has been recognized as an important foundation for software systems. Software architecture has a key influence on the quality of a software-intensive system. It is clear that the overall design of a software-intensive system can be critical in supporting system quality requirements for reliability using redundant components, or for performance using caching components. It is thought that software architecture can be key in the achievement of any kind of system quality.

One important system quality enabled by software architecture is the ability to readily evolve the system. When a system is expected to undergo extensive evolution after deployment, it can be more important that the system be easily evolvable than that it be exactly correct at first deployment (Maier and Rechtin 2000). More broadly, maintenance cost has a strong connection with software architecture. Typically, the cost of maintaining a system for its

whole cycle is two thirds of the total system cost. In other words, Defence will spend two times more of the initial acquisition cost in years to come for upgrading and enhancing a system's capabilities over its life cycle. A software architecture designed to support maintainability can help to ensure that future upgrades are feasible at reasonable cost. There are Defence examples that project cancellations are due to software problems that can be traced back to limitations of software architecture or to the lack of any high-level assurance activities to ensure that safety critical requirements are satisfied (DOD 2009).

Software provides many important properties and functionalities in a modern complex system, but is just part of the whole system. The system architecture describes the overall system decomposition, including software-intensive subsystems. However, a software architecture will not always appear as a neatly contained subsystem in a system architecture. Often the functionality and system qualities supported by the overall software architecture depend on interactions and relationships between separate system components. Software architecture and system architecture can be inter-dependent. Analysis of a software architecture will be based partly on assumptions about the system architecture, to show how system qualities are satisfied. Analyses of software architectural decisions can be used to assess design trade offs for meeting system qualities such as performance, modifiability, usability, or security.

3 Software Architecture Evaluation

3.1 What is Software Architecture?

Australian Defence reports refer to the definition of software architecture in the IEEE 1471-2000 standard (later in ISO/IEC 42010 Architecture Description standard (ISO 2011)): "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution".

Software architecture represents system software structure in terms of software components, how they interact, and how they support various quality requirements such as performance, security, or reliability. It also defines system-wide design rules and considers how a system may adapt to new requirements or environments.

3.1.1 Software Architecture Defines Structure

Much of a software architect's time is concerned with how to sensibly partition a software system into a set of inter-related sub-systems, components, or modules. Different system requirements and constraints will define the precise meaning of "sensibly" – an architecture must be designed to meet the specific requirements of the system it is intended for, under the system's environment constraints.

For example, a security requirement for a system may restrict where data can be stored and how it is communicated, but a user access requirement might require that some services are accessible from mobile devices. Both of these impose some structural constraints on system

design but simultaneously leave open many design options for partitioning functionality across a collection of related components.

The software architecture describes how responsibilities are assigned to each constituent software component. These responsibilities define the tasks a component must perform within the software system. Each component plays a specific role, and the overall component ensemble that comprises the architecture provides all the required functionality.

A key structural issue for nearly all system design is minimizing dependencies, creating a loosely coupled architecture from a set of highly cohesive sub-systems or components. A dependency exists between systems when a change in one potentially forces a change in others. By eliminating unnecessary dependencies, changes are localized and do not propagate throughout an architecture. This is critical for system integration and affects the ability for components to be independently upgraded later without affecting others. Excessive dependencies are simply a bad thing. They make it difficult to make changes to systems, more expensive to test changes, they increase integration costs, and make concurrent, team-based development harder. This is especially so when there is only limited control over possible modifications, for example when dependent components are third-party OTS systems, legacy systems or systems belonging to other organisations.

3.1.2 Software Architecture Addresses Non-functional Requirements

Non-functional requirements are concerned with *how* a system provides required functionality. There are three distinct areas of non-functional requirements:

Technical constraints: These constrain design options by specifying certain technologies that the system must use. "We only have Java developers, so we must develop in Java." "The existing database is Oracle."

Business constraints: These constrain design options, but for business, not technical reasons. For example, "In order to conduct a joint operation, we must interface with XYZ systems." Another example is "The supplier of our current OTS system has raised prices prohibitively, so we're moving to another system."

Quality attributes: These define a system's requirements in terms of scalability, availability, ease of change, portability, usability, performance, and so on. Quality attributes address issues of concern to system users, as well as other stakeholders like the project team itself or the project sponsor.

For a software-intensive system, the software architecture plays an important role in achieving these non-functional requirements, and we need software architecture evaluation to determine whether the architecture has indeed addressed them adequately.

Software architecture also provides the foundation for software reuse, OTS component use, and integration with other systems. Software architecture has a profound influence on the project organizations' functioning and structure. Architecture is the key framework for major technical decisions and many business decisions.

3.2 What is Software Architecture Evaluation?

Software architectures are high-level designs. The aim of software architecture evaluation is to increase the confidence of the evaluation team that a software architecture is fit for its purpose. Evaluation has to be achieved expeditiously, as the detailed design, implementation or acquisition cannot generally commence until the architecture is agreed. The goal is to be as rigorous and efficient as possible, given the limited level of detail available on the proposed design. Validating the proposed system is challenging because whether for a new system, or for the evolution of an existing system, the proposed system does not yet exist. The design will also likely include new systems that have yet to be built, and black box OTS systems and existing operational systems that must be integrated. So, the system cannot be executed or tested to determine whether it fulfils its requirements. Also, for software systems, there are few precise and accurate design analysis techniques, and approximate analyses are not always safe. Nonetheless, an assessment must be made about whether all the proposed parts in the design can be made to work together in the planned way to fulfil the functional and non-functional (quality) requirements.

3.2.1 Architecture Evaluation Methods

Abowd (Abowd, Bass et al. 1997) proposed two broad categories of software architecture evaluation. Questioning approaches focus on qualitative analysis, and include scenario-based analysis techniques. Measurement approaches are quantitative in nature, and include metrics and simulation techniques.

The architecture evaluation research community has developed various methods such as Scenario-Based Architecture Analysis Method (SAAM) and Architecture Tradeoff Analysis Method (ATAM) (Kazman, Klein et al. 2004), Architecture Level Maintainability Analysis (ALMA) (Bengtssona, Lassingb et al. 2004) and Performance Analysis of Software Architecture (PASA) (Williams and Smith 2002). There have been several comparative studies of these methods (Dobrica and Niemela 2002; Babar, Zhu et al. 2004). The architecture evaluation field is reasonably mature.

Scenario-based architecture evaluation methods such as SAAM and ATAM are widely used. Scenario-based methods were developed at the Software Engineering Institute (SEI) to tease out non-functional requirements concerning a software architecture through manual evaluation, prototyping or high-level testing.

A scenario is a hypothetical but reasonable circumstance that may be faced by the system. Scenarios are related to architectural concerns such as quality attributes, and they aim to highlight the consequences of the software architectural decisions that are encapsulated in the design. Operational scenarios can be used to investigate non-functional requirements such as for system performance or reliability, and development or maintenance scenarios can be used to investigate non-functional requirements such as for modifiability or interoperability. The various scenario-based methods involve working out how the system is to respond to this stimulus. If the response is desirable, then a scenario is deemed to be satisfied by the software architecture. If the response is undesirable, or hard to quantify, then a flaw or an area of risk in the architecture may have been uncovered.

Scenarios have been used for a long time in several disciplines including military and business strategy. The software engineering community uses scenarios in user-interface engineering, requirements elicitation, performance modelling and in software architecture evaluation (Kazman, Abowd et al. Nov. 1996). Scenarios are effective for architecture evaluation because they are very flexible; scenarios are used for evaluating most quality attributes. For example, we can use scenarios that represent component failure to examine availability and reliability, scenarios that represent change requests to analyse modifiability, scenarios that represent threats to analyse security, or scenarios that represent ease of use to analyse usability. Scenarios are normally very concrete, enabling the user to understand their detailed effect (Lassing, Rijsenbrij et al. 2003).

The software architecture evaluation community has developed and assessed a number of techniques, both top-down and bottom-up, for eliciting a suitable set of scenarios that is as complete as possible. Regardless of the technique used to elicit scenarios, the measure of completeness is with respect to the business goals. Completeness of a set of scenarios is determined by finding out whether or not there is a business goal that is not covered by some scenarios. If there are some business goals not covered by the gathered scenarios, then the scenarios are incomplete. Equally important is whether a scenario cannot be mapped to a business goal, which may indicate that the statement of business goals is incomplete.

In an ideal world, the architecturally significant requirements for a system to be reviewed would have been completely and unequivocally specified in a requirements specifications document, evolving ahead of or in concert with the architecture specification. In reality, requirements documents sometimes do not properly address architecturally significant requirements. Specifically, the requirements for both existing and planned systems are often missing, vague, or incomplete. Typically the first goal of an architecture analysis is to elicit the specific quality goals against which the architecture will be judged.

Table 1 Six elements scenario generation framework (Bass, Bachmann et al. 2003)

Elements	**Brief Description**
Stimulus	A condition that needs to be considered when it arrives at a system
Response	The activity undertaken after the arrival of the stimulus
Source of Stimulus	An entity (human, system, or any actuator) that generates the stimulus
Environment	A system's condition when a stimulus occurs, e.g, overloaded, running etc.
Stimulated Artifact	Some artifact that is stimulated may be the whole system or a part of it.
Response measure	The response to the stimulus should be measurable in some fashion so that the requirement can be tested.

The scenario generation framework shown in Table 1 is considered effective for eliciting and structuring information about quality-sensitive scenarios gathered from stakeholders or distilled from secondary sources of architecture knowledge such as design patterns. It has been argued (Kazman, Klein et al. 2004) that this framework provides a relatively rigorous

and systematic approach to capture and document general scenarios, which can be used to develop concrete scenarios and to select an appropriate reasoning framework from which to evaluate software architecture.

Once the scenarios have been defined, the evaluation methods for different quality attributes vary. Some are very quantitative such as for performance (Williams and Smith 2002). Some are qualitative techniques such as for modifiability (Bengtssona, Lassingb et al. 2004). There are also methods that quantitatively deal with the trade-offs made among different quality attributes and the sensitivity of architecture decisions (Zhu, Aurum et al. 2005).

In order to deal with black-box OTS software, specific techniques were developed to select OTS components in the context of architecture evaluation. These methods include COTS mismatch resolution (Abdallah Mohamed 2008) and architecture-driven OTS selection (Liu and Gorton 2003). Some techniques propose new ways of judicially exercising parts of the black-box OTS system for specific application profile to gain performance insights into the future system (Zhu, Liu et al. 2007).

There are various practical architecture measurement concepts that are not necessarily linked to specific non-functional requirements or scenarios. Some general techniques have been developed to measure the level of reuse, number of complex interfaces, number of underused interfaces, trends, architecture-related work progress, architecture description completeness, consistency, and general metrics for structural cohesion and coupling (Bianchi 2012). Although these are not specific to individual quality attributes or application requirements, they can often be used as early indicators of architecture characteristics.

3.2.2 International Standards on Architecture Evaluation

Industry best practices are often formally standardised. The maturity and adoption of architecture evaluation is reflected in two new international standards related to architecture description and evaluation. The ISO/IEC 42010 standard on architecture description (ISO 2011) was officially published in 2010. The ISO/IEC 42030 standard on architecture evaluation (ISO 2012) is currently being developed with the first working draft being circulated. The final ISO/IEC 42030 standard is to be published in 2013.

It is important to note that Australia is among the half-dozen countries actively developing the two ISO standards; not just providing comments and casting votes. Authors of this report are the Australian representatives to the working group responsible for the two standards. In the past, early access to the ISO/IEC 42010 (Architecture Description) standard has been provided to Australia Defence and feedback has been incorporated into the final version of the standard. Australia Defence has a basis on which to encourage the use of these ISO standards both internally and externally.

3.2.2.1 ISO/IEC 42010 (Architecture Description)

Figure 1 illustrates the core concepts in this architecture description standard.

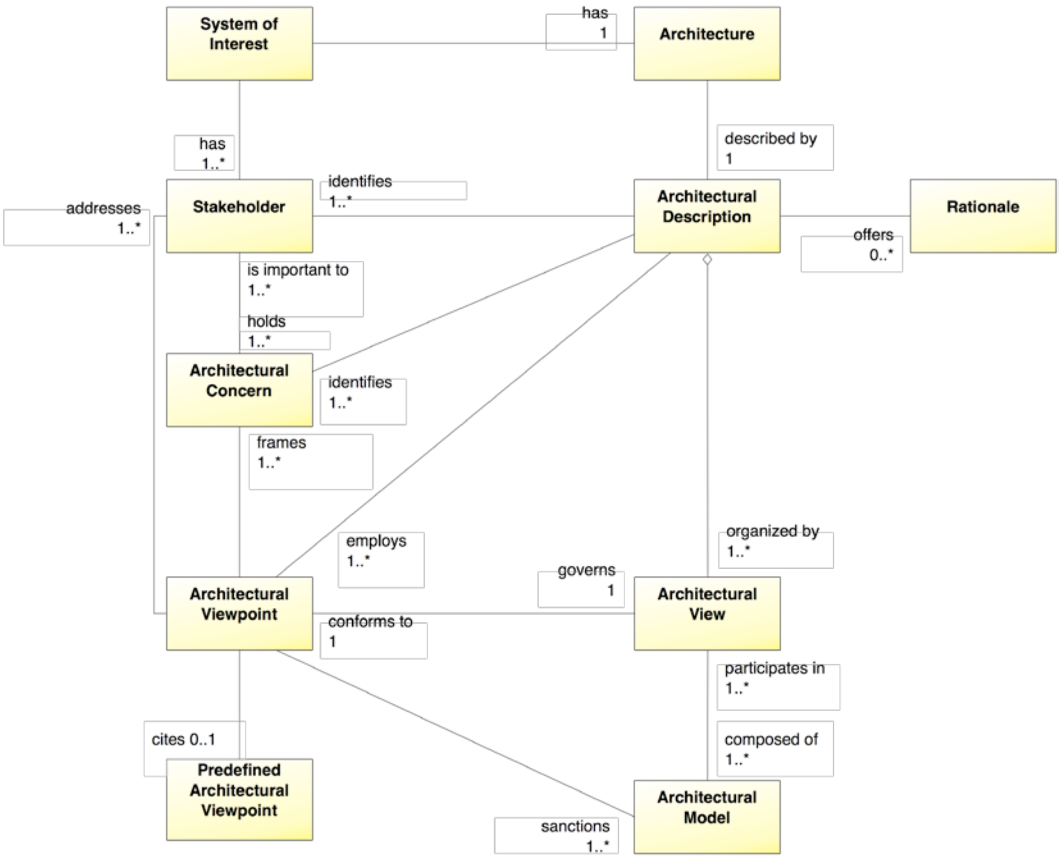

Figure 1 Core Concepts in Architecture Description Standard (reproduced from ISO/IEC 42010)

An **architectural viewpoint** defines the conventions for the construction, interpretation and use of an architectural view to frame an identified quality attribute. An architecture viewpoint will describe: the quality attribute framed by the viewpoint; the modelling languages, notations, rules, constraints, modelling techniques, or analytical methods to be used in constructing a view and its architectural models; consistency and completeness checks; and evaluation or analysis methods and required analysis models.

An **architectural view** is a representation of a whole system from the perspective of an identified set of architecture-related concerns. An architectural description contains one or more architectural views. The requirement that architectural views depict the whole system is essential to the complete allocation of quality attribute assessments within an architectural description. Architectural models, within views, can be used to selectively present portions of the system to highlight points of interest.

An architectural view consists of one or more **architectural models**. Architectural models provide a mechanism to modularize architectural views. In the ordinary case, a view consists of exactly one model. However, there are cases when a view may need to use more than one language, notation or modelling technique to address all of the concerns assigned to it. To do this, a view may consist of multiple architectural models.

A **view correspondence** records a relation between two architectural views. A view correspondence may be used to capture a consistency relation, a traceability relation, a constraint or obligation of one view imposed upon another, or other relations relevant to the architecture being described.

An **architectural rationale** (or simply, rationale) is an explanation or justification for an architectural decision. A rationale explains the basis for the decision, the reasoning that led to the decision, the trade-offs that were considered, the impact of the decision including its pros and cons, and points to further sources of information.

An **architectural decision** (or simply, decision) is a choice that addresses one or more architecture-related quality attributes, and affects (directly or indirectly) the architecture. A decision may address (in full or partial) more than one quality attribute. A decision may affect the architecture in several ways: existence of an architectural entity or the property of some architectural entities. In most cases a decision provides a trace between architectural elements and quality attributes. A decision may raise additional concerns.

Using ISO/IEC 42030, a potential supplier can prepare an architectural description to be used throughout the life cycle to make predictions of the fitness-for-use of a system whose architecture complies with the architectural description or to assess how well architecture-related concerns have been addressed. The architectural description will typically evolve throughout the life cycle as understanding of the architecture grows, and can serve as a means for assessing changes to the system.

Use of Architecture Frameworks

An architecture framework establishes a common practice for creating, organizing, interpreting and analysing architectural descriptions used within a particular domain of application or stakeholder community. For example, DODAF and MODAF are two architecture frameworks used in the US DoD. Australia has a similar AUSDAF for ICT-related defence systems. An architecture framework identifies a set of generic architecture-related concerns (quality attributes), generic stakeholders holding these concerns, and one or more predefined architectural viewpoints which frame these concerns. Architectural frameworks use generic stakeholders to motivate the concerns in the predefined viewpoints defined by the architectural framework. In a predefined viewpoint, a generic stakeholder is identified to establish concerns that the predefined viewpoint frames. As part of applying the framework, generic stakeholders and generic concerns are instantiated within an actual architectural description.

3.2.2.2 ISO/IEC 42030 (Architecture Evaluation)

This standard specifies the practices and products for planning, executing and documenting architecture evaluations for the purpose of

- Determining the quality of an architecture,
- Verifying that an architecture addresses stakeholders' concerns, or
- Supporting decision-making informed by the architecture of the System of Interest.

Figure 2 shows the parties involved in an architecture evaluation. The evaluator is often an independent expert outside the project team.

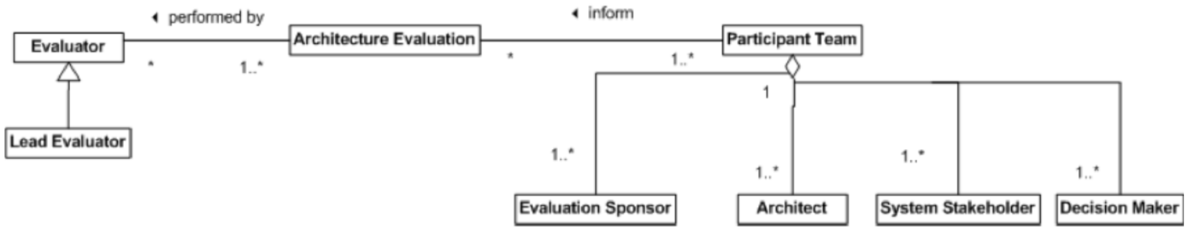

Figure 2 Parties Involved in Architecture Evaluation (reproduced from ISO 42030 Working Draft)

Architecture evaluations can be scoped to assess fulfilment of all or a subset of the concerns. Based on the scope and complexity of the architecture, the evaluation is performed by an architecture evaluator or a team of evaluators. Depending upon the context of the evaluation there may also be third-party evaluators. Multi-member evaluation teams will need a leader, who plans, executes and manages the entire evaluation.

An architecture evaluator can perform architecture evaluation on behalf of a client. Architecture evaluations are commonly initiated either by the acquirer or as a joint acquirer-supplier review. The initiator becomes the evaluation sponsor and outlines the drivers for this exercise.

The evaluation sponsor will provide the purpose for evaluating the architecture. The evaluation team will elaborate the scope and objectives, determine evaluation criteria, determine analysis techniques and tools to use, collect and understand required information, assess the architecture, formulate findings and recommendations, communicate results of the evaluation, manage the architecture evaluation with a plan.

Architecture evaluations are performed for specific purposes or concerns, within a fixed timeframe, by one or more evaluators using various sources of information (human or others). Architecture evaluations need to be managed and governed with a plan.

Figure 3 depicts the concepts relating to the architecture evaluation plan and key contents of the plan. In addition to evaluation criterion, scope, schedule and resources for completing the

evaluation, the plan clearly states how the scope of evaluation will be covered in the form of one or more modules of work. In the context of architecture evaluation, these modules are referred as assessment modules.

Assessment modules are built around or use a specific assessment method. They aggregate the assessment method, desired set of participant roles, the time and support expected from them, any special skills required in the evaluator, the kinds of input information and, artefacts or roles that act as the sources of these information. Architecture viewpoints and architecture frameworks may recommend assessment modules for use by an architecture evaluation. The selection of assessment modules should be influenced by analysis and evaluation methods specified in viewpoints or frameworks that are part of the architecture and architecture description.

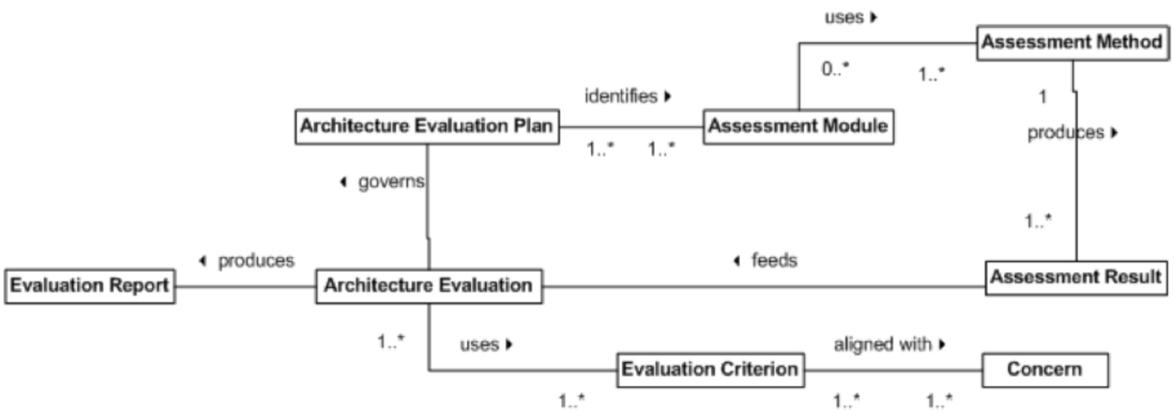

Figure 3 Architecture Evaluation Plan (reproduced from ISO42030 Working Draft)

The assessment module works as a template that can be reused when creating work units for architecture evaluations performed in the enterprise. When an architecture evaluation is in progress the evaluation plan instantiates the planned assessment modules. The actual instances of the assessment module will have the actual artefacts and named resources.

The evaluation criterion is designed keeping the evaluation purpose or concerns in context. Using the evaluation criterion, outcomes and findings of the individual assessment modules are aggregated to draw conclusions or provide recommendations that are documented as part of the evaluation report.

ISO/IEC 42030, once published, will provide an internationally-recognised reference point for conducting architecture evaluation.

4 Software Architecture Evaluation in Industry and Defence

4.1 Software Architecture Evaluations in Industry

There have been many experience reports and case studies published on architecture evaluation. Most notably, AT&T and sub-companies have reported around 700 architecture evaluations between 1998 and 2005. They estimate that projects of 100,000 non-comment source lines of code have saved an average of US$1 million each by using software architecture evaluations for identifying and resolving problems early (Maranzano, Rozsypal et al. 2005).

The primary goal of AT&T's architecture evaluation was to find design problems early in development when they are less expensive to fix. But a number of other key benefits were also identified:

- Leverage experienced people by using their expertise and experience to help other projects in the company
- Let companies better manage software component suppliers
- Provide management with better visibility into technical and project management issues
- Generate good problem descriptions by having the review team critique them for consistency and completeness
- Rapidly identify knowledge gaps and establish training in areas where errors frequently occur (for example, creating a company-wide performance course when many reviews indicated performance issues)

They found architecture reviews could get management attention without personal retribution. Because independent experts conduct reviews and the organization's management sanctions the reviews, they are a safe way to identify a project's most serious issues. Business leaders can also learn more about the architecture and reasons for key technical decisions.

They also found architecture reviews assist organizational change. Architecture reviews provide an intervention opportunity to an organizational change effort. By observing both what projects are doing and how they do it, interactions among the various process participants can introduce change into individual projects and the larger organization.

When conducting architecture evaluations, AT&T identified five key principles for success:

1. **A clearly defined problem statement drives the system architecture**. Problem statements constrain solutions through considerations or constraints on functionality, structure, environment, and cost and time to develop and deploy the solution.

2. **Product line and business application projects require a system architect at all phases**. The system architect, whether a person or small team, is an explicitly identified role. The architect must explain the reasons for the technical decisions that define the system, including alternatives considered, and must document choices in a concise, coherent architecture specification.

3. **Independent experts conduct reviews**. The chosen reviewers are recognized experts in their fields, generally with considerable experience. To assure impartiality, they are also independent of the particular project under review.

4**. Reviews are open processes**. Thus, the review team conducts the review openly and invites project members to attend. Everyone involved in the review process can see the architecture's issues and strengths and note them.

5. **Companies conduct reviews for the project's benefit**. Reaction to the issues discovered, such as a project's architectural changes, replanning, or even cancellation, is the project team and management's responsibility.

Many of these lessons we believe are applicable to Australia defence acquisition projects.

Another more recent survey of the state of practice in architecture evaluation among 235 industry participants identifies a range of benefits, among which the top five are architecture evaluation's ability to identify project risks (88% responses), assess quality (77% responses), identify reuse opportunities (72% responses), promote good practices within an organisation (64% responses) and reduce project cost by reducing design defects (63% responses) (Babar and Gorton 2009). The full list is shown in Table 2.

Table 2 Benefits/Goals of Architecture Evaluation (Babar and Gorton 2009)

Benefits/goals of conducting architecture review	Responses
a. Identifying potential risks in the proposed architecture	76 (88%)
b. Assessing quality attributes (for example, scalability, performance)	66 (77%)
c. Identifying opportunities for reuse of architectural artifacts and components	62 (72%)
d. Promoting good architecture design and evaluation practices	55 (64%)
e. Reducing project cost caused by undetected design problems	54 (63%)
f. Capturing the rationale for important design decisions	51 (59%)
g. Uncovering problems and conflicts in requirements	51 (59%)
h. Conforming to organization's quality assurance process	47 (55%)
i. Assisting stakeholders in negotiating conflicting requirements	37 (43%)
j. Partitioning architectural design responsibilities	34 (40%)
k. Identifying skills required to implement the proposed architecture	34 (40%)
l. Improving architecture documentation quality	34 (40%)
m. Facilitating clear articulation of nonfunctional requirements	27 (31%)
n. Opening new communication channels among stakeholders	27 (31%)

Another industry survey on risk management in architecture evolution also clearly shows that early architecture risk identification leads to better project outcomes and cost savings (Slyngstad, Conradi et al. 2008).

4.2 Software Architecture Evaluation in US DoD System Acquisition

The most notable use of software architecture evaluation in the defence domain is from US DoD. US DoD considers software architecture critical to the quality of a software-intensive system (Bergey and Morrow 2005). For an acquisition organization, such as the US Department of Defense (DoD), the ability to evaluate software architectures early in an acquisition was identified to have a favourable impact on the delivered system (Bergey and Fisher 2001). Software architecture can help mitigate many of the technical risks associated with software-intensive system development, thereby improving the ability of the acquisition to achieve the stated system objectives. In an acquisition context, these architecture evaluations provide a proactive means of gaining early visibility into critical design decisions that will drive the entire system-development effort. They can be performed before a system is selected and integrated to determine if the system can be made to satisfy its desired qualities. Data gathered over multiple years confirmed that the use of architecture evaluations are generally beneficial to system acquisitions and suggests that maximal benefit is achievable only if architecture-centric practices are built into the acquisition process (Nord, Bergey et al. 2009).

4.2.1 The CLIP Example from US DoD

One example from US DoD is the Common Link Integration Processing (CLIP) program which was a US$275 million project. It involved cooperative Air Force/Navy programs with integrated Tactical Data Links (TDLs) across platforms with a TDL requirement. It needed to provide message processing, gateway functionality, and a common interface to enable transition of new and legacy platforms to a Network Centric Warfare (NCW) environment.

The program had some architecture challenges such as:

- Incremental acquisition supporting different platform integration needs and their expected date of integration
- Developing software assets which would be portable to the different reusable platforms using diverse hardware and software
- Integration of CLIP with other DoD systems under development
- Development of a common host interface

Two key architecture evaluation activities were used in various stages of the project. One is the Quality Attribute Workshop (QAW) method that is essentially a quality-attribute-related scenario elicitation activity involving key stakeholders. The other is the ATAM that can be performed relatively quickly and inexpensively. ATAM involves project decision-makers, other stakeholders including managers, developers, maintainers, testers, re-users, end users, customers, and an architecture evaluation team. ATAM seeks to ask questions to uncover:

- Risks: architecture decisions that might create future problems for some quality attribute
- Sensitivity points: properties of one or more components (and/or component relationships) that are critical for achieving a particular quality attribute response (i.e., where a slight change in a property can make a significant difference in a quality attribute)
- Tradeoffs: decisions affecting more than one quality attribute.

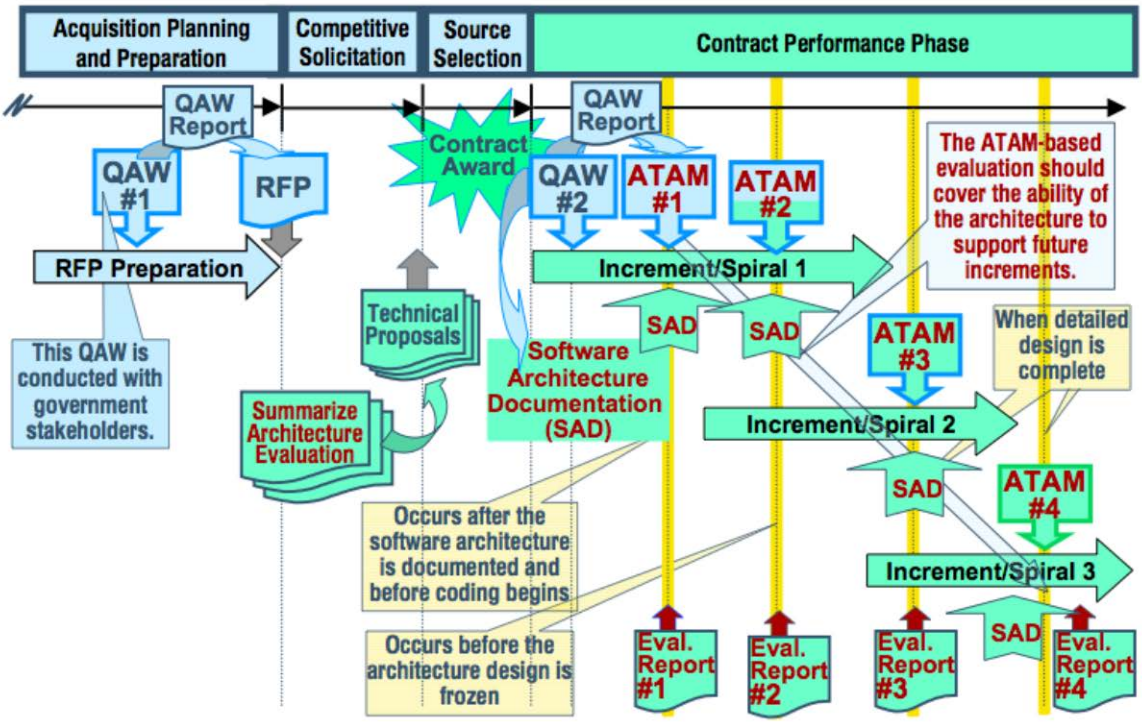

Figure 4 QAW and ATAM Integration (reproduced from Bergey and Morrow 2005)

Figure 4 illustrates the integration. A QAW was conducted during the acquisition planning stage and all architecture related requirements were integrated into the RFP. Thus enough information on architecture was available inside supplier responses for source selection. After the contract award in May 2005, one more QAW and four ATAMs were conducted to provide a common forum for discussing quality attribute requirements and architectural decisions and to gain stakeholder buy-in. Software architecture documentation was delivered in support of Preliminary Design Review (PDR) and another ATAM engagement in Nov 2005 to successfully increase communication among stakeholders, clarify quality attribute requirements and identify software risks early in the development cycle.

4.2.2 Integrating Software Architecture Evaluation in US DoD Acquisition

From the CLIP example, generalisations were made to identify ways of integrating architecture evaluation into the acquisition process.

In US DoD, acquisition planning precedes the entire solicitation process and includes generating and validating product requirements (e.g., functional and quality requirements such as reliability or performance). In the pre-award phase, a solicitation package is developed. It informs potential suppliers what the requirements of the acquisition are, how to prepare their proposals, how proposals will be evaluated, and when to submit their proposals. Solicitation packages take various forms and are referred to differently. However, they all have the same characteristics noted here. A Request For Proposal (RFP) is used here to represent all such packages. These are similar to Australia Defence's Request For Tender

(RFT) package which includes the Function and Performance Specification (FPS), Concept Document (OCD), Test Concept Document (TCD) among others.

Software architecture and software architecture evaluation requirements are integrated explicitly in different parts of the RFP. For example, Section C in DoD's RFP contains supplier work requirements in the form of a statement of objectives (SOO) or statement of work (SOW) along with product requirements such as a system performance specification (containing functional and quality requirements). A US DoD system specification typically has two main sections of interest. Section 1 specifies functional and quality requirements for the system. Here, quality requirements refer to those quality attributes of the system and their respective characterizations. Modifiability, reliability, and security are examples of the types of system quality attributes that may be considered. For example, if reliability is a required quality attribute, a characterization might be that "the system will not fail under maximum load conditions". Eliciting the quality attributes of primary interest as well as their characterizations for the system in question are part of the ATAM. Section 2 of the system specification describes the software architecture evaluation methods, such as the ATAM, to be used in determining if the software architecture can support the satisfaction of the requirements in Section 1.

All of the key US DoD acquisition documents are architecture driven, including the Acquisition Strategy and Acquisition Plan, System Engineering Plan, Test and Evaluation Master Plan, Request for Proposal, Statement of Work and System Requirements Document.

For example, Section C of the RFP regarding Statement of Work, explicitly says "An evaluation team shall conduct a series of software architecture evaluations in accordance with the special requirements of section H". Section H of RFP (special contract requirements) includes the detailed requirements specifying how the software architecture evaluation are to be conducted using the ATAM. Section J of RFP (Contract deliverables requirements list), includes "software architecture documentation" and "the software architecture evaluation report"

Section L (Proposal Preparation Instructions) describes what potential suppliers should address in their proposals and the response that is required. Typically, the acquirer would ask the potential suppliers for responses in several volumes, such as a technical volume, past performance volume, management volume, and cost volume. There are no set rules for what these volumes exactly contain. In the technical volume, an acquirer may ask potential suppliers to describe their proposed approach for implementing the software architecture requirements and performing an architecture evaluation. In the past performance volume, an acquirer may ask suppliers to describe previous work on software architecture development and architecture evaluation.

Section M (Evaluation Factors for Award) tells potential suppliers how their proposals will be evaluated. This typically includes specifically what areas of the supplier's proposed approach are to be evaluated as part of the proposal evaluation and the specific criteria to be used for judging the supplier's proposed approach to meeting the RFP/contract requirements for these factors. To incorporate architecture evaluation, Section M must specify how the architecture

evaluation will relate to the factors. And, it must specify the criteria to be used in judging the bidder's approach to satisfying the RFP/contract architecture requirements.

Figure 5 and Figure 6, reproduced from (Bergey and Fisher 2001; Bergey and Morrow 2005), shows some integration opportunities division of tasks between acquirers and suppliers.

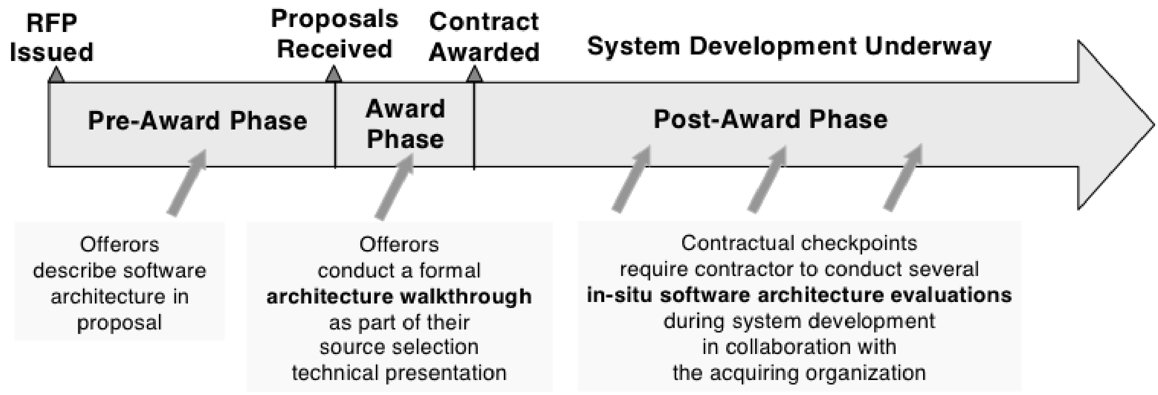

Figure 5 Opportunities for Conducting Architecture Evaluation during Acquisition

During the post-award phase, making architecture evaluation a contractual checkpoint is also an effective way to gain insight into the architecture's ability to satisfy system requirements early in its development and integration. Such an evaluation can help the acquirer and supplier to select an architecture decision among several candidate decisions and to surface/mitigate risks associated with architectural decisions early.

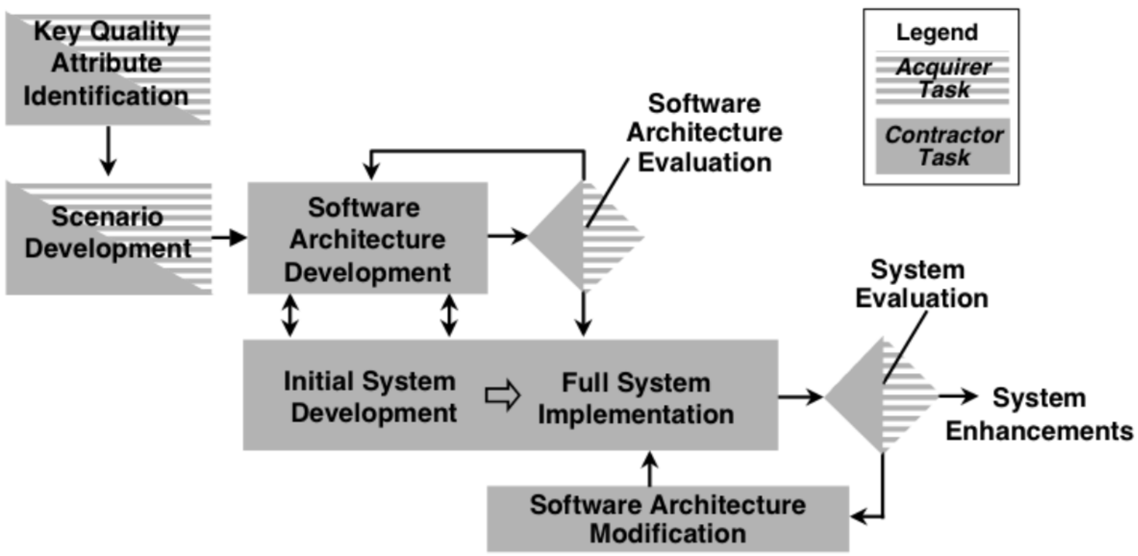

Figure 6 Key Acquirer and Contractor Tasks

In US DoD, the following must be addressed for software architecture evaluation requirements:

- What evaluation method is to be used and what are the steps?
- Who are the participants in the architecture evaluation?
- What are their roles and responsibilities?
- How many evaluations need to be conducted and when?
- If multiple evaluations are involved, how are they to be staged?
- What are the prerequisites for conducting the evaluations?
- What is involved in terms of time, effort, and cost?
- How are evaluation team responsibilities to be transitioned?
- How will the objectivity of the participants be ensured?
- How are the evaluation results to be captured and used?
- What contract deliverables need to be included?
- How can the evaluations be carried out collaboratively to ensure both government and contractor stakeholders play an active role?
- What training will be provided for the evaluation team members?

A proven means achieving this (Bergey 2009) is to provide a sample Software Architecture Evaluation Plan that a DoD program office can easily customize and use in its own Request for Proposal (RFP)/contract. The sample plan covers all aspects on the "who, why, when, where, and how" of the government's approach to conducting a software architecture evaluation.

We observe many of the lessons here are applicable to the Australian Defence acquisition process.

5 The Need for Software Architecture Evaluation in Acquisition of Software-Intensive Systems in Australian Defence

The life cycle process for Australian Defence's capability system consists of several major phases: needs, requirements, acquisition, in service and disposal. Our main focus is on the requirements and acquisition phase especially when DMO is involved.

There are a number of documents that will be created in preliminary, draft and final form such as Request for Information (RFI), Request for Proposal (RFP), Material Acquisition Agreement, Technical Risk Assessment, Request for Tender Package which includes the Operational Concept Document (OCD), Function and Performance Specification (FPS), Test Concept Document (TCD) among others. Some are more important in the source selection and solicitation stage before awarding the contract. Some are still very valid after the contract award when conducting various system reviews and when transitioning the materiel and capabilities into service.

As we can see from the US DoD example, architecture evaluation related information request/response and activities can be integrated across the lifecycle. We note that many acquisition projects in Australian defence acquire MOTS, COTS, GOTS and other OTS systems whenever possible. This is perhaps more so than the US DoD. Although some architecture decisions in OTS are more difficult to change, there are still many key architecture decisions need to be changed or made during integration and customisation.

There is already some recognition in Defence about the importance of architecture generally. For example the 2009 NCW roadmap paper outlines the need of a single Enterprise Architecture to support the networked force build is fundamental to achieving the necessary collaboration and synchronisation. A key aspect of architectural design is the development and use of architecture framework views to establish a 'common language' between diverse stakeholders, to manage the inherent complexity of system of systems, particularly under the influence of diverse mission requirements, and to enable incremental capability development and integration into the force structure (DoD 2009). The Australian National Audit Office (ANAO)'s report the Collins Class Submarine and Seasprite also highlighted the importance of early design reviews and benefits gained in having a software architecture that allowed for a graceful degradation in performance even if the operator neglected to manage the object density there was no possibility of system malfunction or software stoppage (DOD 2009).

Another initiative between DMO and DSTO has seen the establishment of the Defence Systems Integration Technical Advisory (DSI–TA). This new organisation forms part of the DMO, with support provided by DSTO. The main objective of the DSI–TA is to provide independent identification, assessment and mitigation strategies for 'systems' and 'systems of systems' integration risks in current and future major DCP projects. One of its major tasks is to create integrated defence architecture and sub-domain/section architectures. The quality in these architectures and their applications in individual systems need to be evaluated at different time.

In addition, an explicit architecture also helps future maintenance and upgrade and other suppliers (especially Australian SMEs) provide better integration into key mission systems.

We make the following general recommendations.

Recommendation 1: Incorporate software architectural requirements into the Request for Tender package.

Contract requirements frame the development of a system, so the best way to ensure adherence to a sound software architecture is to stipulate architectural requirements in the system specification. The first principle from the industry experience of AT&T in architecture evaluation is that a clearly defined problem statement drives the software-system architecture**.** The various system capabilities needs should drive the software-system architecture development and be architecture-conscious. It is therefore recommended that software architectural requirements are incorporated into the Request for Tender package.

The FPS (DoD 2011) specifies the formal requirements for the Materiel System and provides the basis for design and Verification of the system. The FPS already provides the vehicle for the capture of formal, verifiable and unambiguous requirements, effectively 'distilled' from the OCD. The FPS is intentionally written using formal language, with all requirements in the FPS traceable to needs identified in the OCD.

In the current FPS, there are already large sections (section 8.4.x) on quality attributes such as availability, reliability, maintainability, deployability, transportability, usability and safety. It also contains information on environmental conditions, adaptation requirements and design and implementation constraints. There is an opportunity here to tease out the architecturally significant requirements and ask suppliers to respond to these requirements using software architecture description and evaluation report.

There is also a special section (Section 3.12) in FPS that outlines the "Architecture, Growth and Expansion" requirements. This section specifies the applicable architectural and other requirements to accommodate the need for flexibility, growth, and expansion for relevant subsets of the Materiel System to support anticipated areas of growth or changes in technology, threat, or mission. Software architecture-driven modifiability analysis is a well-understood field, though not well-established in the Australian Defence context. There is an opportunity to require a more rigorous modifiability analysis based on explicit software architecture descriptions. International standards on architecture description such as ISO/IEC 42030 should be recognised in an appropriate way when defining policies or RFT conditions requesting information about software architectures.

Since the contract requirements drive the entire development of a system, the best way to ensure adherence to a sound software architecture is to stipulate software architectural requirements in the system specification. It is therefore recommended that software architectural requirements are incorporated into the Request for Tender package

Recommendation 2: Integrate software architecture evaluation activities across the acquisition process.

The successful use of software architecture evaluation in the US DoD provides grounds for believing that software architecture evaluation should be useful in Australia. The Australian Defence Architecture Framework (AUSDAF) is an authorised framework of architectural views that can be incorporated into the OCD where they help stakeholders to understand the proposed system from their perspective. Applicable AUSDAF products are integrated into the OCD at relevant steps in the development process. AUSDAF products are currently only applicable to certain types of OCD (i.e. to the ICT elements). We believe that there is an

opportunity to develop other suitable architecture frameworks for all of defence systems beyond the ICT elements.

Architecting a system is one of earliest and hardest activities that the developer has to conduct. Any mistakes that may happen at this stage will ripple down the system implementation if they are not detected and rectified earlier. It is well known that it is approximately few-orders-of-magnitude cheaper to correct any errors at software architecture than to detect and correct them at the testing phase. It is therefore recommended that software architecture evaluations are conducted as a risk mitigation strategy in a software-intensive system acquisition.

Two principles from AT&T's industry experience of successful software architecture evaluation are that reviews should be open processes, and be conducted for the project's benefit. The review team should conduct the architecture evaluations openly with all project members invited to attend. The review process enables all stakeholders to see the architecture's issues and strengths. The project team and management then bear responsibility for the action plans to address the issues discovered, such as architectural changes or project replanning. International standards on architecture evaluation such as ISO/IEC 42030 should be recognised when defining policies on conducting architecture evaluations.

Recommendation 3: Build software architecture capabilities in Australian defence industry.

Defence acquisitions involve international big companies, which often reside overseas and are subject the countries of origin' rules and regulations such as US ITAR. These rules may prevent DMO to engage international experts in software architecture to conduct software architecture evaluations on DMO behalf. It is therefore imperative to develop and sustain software architecture capabilities in Australia Defence industry.

Two principles from AT&T's industrial experience of successful software architecture evaluation are that all projects require a software-system architect at all phases, and that independent experts should conduct reviews. The software-system architect, whether a person or small team, should an explicitly identified role; and the software architecture evaluation team should be led by recognized experts external to the project teams. There is a need for software architecture capability within both DMO and industry.

It is recommended that DMO develops indigenous software architecture capabilities in Australia Defence industry by putting together a government and industry team to develop specification, data item description (DID) template, and contractual language for incorporating into the Request for Tender; to develop a procedure and techniques for evaluating software architectures; and to train and sustain a pool of software architecture evaluators to support DMO in evaluating software architectures.

Recommendation 4: Provide appropriate contractor incentives for software architecture definition, analysis, and evaluation throughout the project lifecycle.

Developing and analysing a proper software architecture design requires time and expertise. This cost may arise even from the initial response to a RFT. Although defining a clear and

appropriate software architecture early in the project may reduce overall project cost and schedule risk, there may nonetheless be further costs to maintain a proper control of the software architecture throughout a project. Contractors may already conduct in-house software architecture design, but currently this activity and any software architecture documentation, if it exists, are not visible to DMO. Where software architectural definition and analysis is currently performed, under pressures of time and budget, contractors may prioritise meeting immediate requirements at the expense of longer-term benefits from maintaining software architectural integrity. It is therefore recommended that contractors are given incentives to establish, maintain, and evaluate software architectures during the life cycle of system development.

6 Conclusion

Australian Defence and the local defence industry are facing new challenges from ever-larger, more complex and interconnected software-intensive systems. Many are based on OTS systems supplied by international primes. Risks need to be better managed in such an environment.

In this report, we described the concept of architecture evaluation, and its state of art and practice in industry. For companies such as AT&T, significant cost was saved (an average of 1 million USD saving for 100K-lines-of-code projects) due to early detection of design problems through architecture evaluation. Architecture evaluation can also socialise best practices within an organisation and provide visibility of system-level impacts from technical decisions to upper management. In the defence context, US DoD has successfully incorporated architecture evaluation activities into different stages of acquisition projects through explicit requirements in both RFPs and contracts.

We also made some general recommendations in the context of Australian defence acquisition, especially in the development of CDD and FPS. More detailed recommendations and implementation strategies require further investigation.

We believe integrating systematic architecture evaluation into Australian Defence acquisition will bring a number of key benefits:

- Early detection of design problems to significantly reduce later rework cost
- Manage various kinds and levels of risks better with key stakeholder involvement
- Help identify reusable components/systems and infrastructure to bring previously separate capabilities into multi-role platforms
- Socialise best practices within Australian Defence and across to industry
- Help build up industry capabilities in architecture and design, which is a key part of knowledge-based economy

- Allow Australian SMEs to integrate better with global supply chain using explicit integration architectures provided by both defence and international primes

- Better communication to all stakeholders at various level through using architecture as the centre manage technical and non-technical risks

7 References

Abdallah Mohamed, G. R. A. E. (2008). "Optimized mismatch resolution for COTS selection." Software Process: Improvement and Practice **13**(2): 157-169.

Abowd, G., L. Bass, P. Clements, R. Kazman, L. Northrop and A. Zaremski (1997). "Recommanded Best Industrial Practice for Software Architecture Evaluation." CMU/SEI, CMU/SEI-96-TR-025

Babar, M. A. and I. Gorton (2009). "Software Architecture Review: The State of Practice." Computer **42**(7): 1-8.

Babar, M. A., L. Zhu and R. Jeffery (2004). A Framework for Classifying and Comparing Software Architecture Evaluation Methods. 15th Australian Software Engineering Conference.

Bass, L., F. Bachmann and M. Klein (2003). "Deriving Architectural Tactics-A Step toward Methodical Architectural Design." Software Engineering Institute, Carnegie Mellon University,

Bengtssona, P., N. Lassingb, J. Boschc and H. v. Vliet (2004). "Architecture-level Modifiability Analysis (ALMA)." Journal of Systems and Software **69**(1-2): 129-147.

Bergey, J. (2009). "A Proactive Means for Incorporating a Software Architecture Evaluation in a DoD System Acquisition." SEI, CMU/SEI-2009-TN-004

Bergey, J. and M. Fisher (2001). "Use of Architecture Tradeoff Analysis Method (ATAM) in the Acquisition of Software-Intensive Systems." Software Engineering Institute (SEI),

Bergey, J., M. Fisher and L. G. Jones (2002). "Use of Architecture Tradeoff Analysis Method (ATAM) in Source Selection of Software-Intensive Systems." Software Engineering Institute (SEI),

Bergey, J. and T. Morrow (2005). "Integrating Software Architecture Evaluation in a DoD System Acquisition." US Defense Technical Information Center (DTIC),

Bianchi, A. (2012). Software and System Architecture Measurement Framework. Practical Software & System Measurement Working Group Meeting

Boehm, B., R. Valerdi and E. Hon (2008). "The ROI of Systems Engineering: Some Quantitative Results for Software-Intensive Systems." Systems Engineering **11**(3): 221-234.

Dobrica, L. and E. Niemela (2002). "A Survey on Software Architecture Analysis Methods." IEEE Transaction on Software Engineering **28**(7): 638-653.

DoD (1985). "Military Standard, Technical Reviews and Audits for Systems, Equipments, and Computer Software (MIL-STD-1521 Revision B)." Defense Acquisition University (DAU),

DoD (1994). "Software Development & Documentation (MIL-STD-498)." Department of Defense (DoD),

DoD (2009). "NCW Roadmap.", Department of Defence, DPS:FEB005/09, Canberra

DOD (2009). The Super Seasprite, Australian National Audit Office (ANAO).

DoD (2010). "Building Defence Capability: a policy for a smarter and more agile defence industry base." Department of Defence, Canberra

DoD (2011). "Defence Capability Definition Documents Guide." Defence Materiel Organisation, DMH(ENG) 12-3-003

DoD (2011). "Function And Performance Specification (FPS) Development Guide." Defence Materiel Organisation, DMH(ENG) 12-3-005

Ferguson, J. (2001) "Crouching Dragon, Hidden Software: Software in DoD Weapon Systems. IEEE Software **18**(4): 105-107.

Hantos, P. (2004). System and Software Reviews since Acquisition Reform. Southern California Software Process Improvement Network (SPIN).

ISO (2011). ISO/IEC/IEEE 42010:2011 Systems and Software Engineering — Architecture Description, International Standard Organisation.

ISO (2012). ISO/IEC/IEEE 42030 Systems and Software Engineering — Architecture Evaluation (Working Draft 1), International Standard Organisation.

Kazman, R., G. Abowd, L. Bass and P. Clements (Nov. 1996). "Scenario-Based Analysis of Software Architecture." IEEE Software.

Kazman, R., M. Klein and P. Clements (2004). "ATAM: Method for Architecture Evaluation." Software Engineering Institute, Carnegie Mellon University,

Lassing, N., D. Rijsenbrij and H. v. Vliet (2003). "How Well can we Predict Changes at Architecture Design Time?" Journal of Systems and Software **65**(2): 141-153.

Liu, A. and I. Gorton (2003). "Accelerating COTS midellware Acquistion: The i-Mate Process." IEEE Software **20**(2): 72-79.

Maier, M. W. and Rechtin E. (2000). "The Art of Systems Architecting." RCR Press

Maranzano, J. F., S. A. Rozsypal, G. H. Zimmerman, G. W. Warnken, P. E. Wirth and D. M. Weis (2005). "Architecture Reviews: Practice and Experience." IEEE Software **22**(2): 34-43.

Nord, R., J. Bergey, J. Blanchette, Stephen and M. Klein (2009). "Impact of Army Architecture Evaluations " SEI, CMU/SEI-2009-SR-007

Slyngstad, O. P. N., R. Conradi, M. A. Babar, V. Clerc and H. v. Vliet (2008). Risks and Risk Management in Software Architecture Evolution: An Industrial Survey. Asia-Pacific Software Engineering Conference, Beijing, China.

Smith, J., G. Egglestone, P. Farr, T. Moon, D. Saunders, P. Shoubridge, K. Thalassoudis and T. Wallace (2004). "Technical Risk Assessment of Australian Defence Projects " DSTO,

Williams, L. G. and C. U. Smith (2002). PASA: An Architectural Approach to Fixing Software Performance Problems. 28th International Computer Measurement Group Conference, Reno, Nevada, USA.

Zhu, L., A. Aurum, I. Gorton and R. Jeffery (2005). "Tradeoff and Sensitivity Analysis in Software Architecture Evaluation Using Analytic Hierarchy Process." Software Quality Journal **13**(4): 357-375.

Zhu, L., Y. Liu, I. Gorton, N. B. Bui and R. Jeffery (2007). "MDABench: Customized Benchmark Generation Using MDA." Journal of Systems and Software **80**(2): 265-282.

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA					1. DLM/CAVEAT (OF DOCUMENT)

2. TITLE	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)
The Need for Software Architecture Evaluation in the Acquisition of Software-Insentive Sysetms	Document (U) Title (U) Abstract (U)
4. AUTHOR(S)	5. CORPORATE AUTHOR
Liming Zhu, Mark Staples and Thong Nguyen	DSTO Defence Science and Technology Organisation 506 Lorimer St Fishermans Bend Victoria 3207 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TR-2936	AR-015-849	Technical Report	January 2014

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
	04/245	DSTO	27	35

13. DSTO Publications Repository	14. RELEASE AUTHORITY
http://dspace.dsto.defence.gov.au/dspace/	Chief, Aerospace Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS

Software architecture; software evaluation

19. ABSTRACT

The software architecture for a software-intensive system defines the main elements of the system, their relationships, and the rationale for them in the system. Software architecture is fundamental to whether a system can achieve its quality objectives. Architecture evaluation is an approach for assessing whether a software architecture can support the system needs, especially its non-functional requirements (also known as quality requirements). Architecture evaluation can be used at different stages of a project, and is an effective way of ensuring design quality early in the lifecycle to reduce overall project cost and to manage risks.

This report describes software architecture and architecture evaluation, and summarises some of the key benefits for software architecture evaluation that have been observed both in industry and in international Defence contexts. We make some general recommendations about architecture evaluation in the context of Australian defence acquisition.

Page classification: UNCLASSIFIED